10004 UN CASO PRÁCTICO DE UTILIZACIÓN DEL SIMULADOR NS-2 EN EL ANÁLISIS DEL COMPORTAMIENTO DE VARIANTES DEL PROTOCOLO TCP CON FINES DE INVESTIGACIÓN Y ENSEÑANZA

Diego R. Rodríguez Herlein[(1)(2)], Carlos A. Talay[(1)(3)], Claudia N. González[(1)(4)], Franco A. Trinidad[(1)(5)], Luis A. Marrone[(6)]

[(1)]Licenciatura en informática, UNPA-UARG

Río Gallegos, Argentina Campus universitario - Oficina B 18

[(2)]diegorh@gmail.com

[(3)]carlostalay@yahoo.com.ar

[(4)]claudiagonzalez@yahoo.com.ar

[(5)]talejandro.franco@gmail.com

[(6)]L.I.N.T.I. – Universidad Nacional de La Plata La Plata, Argentina

Calle 50 y 115 – 1er. Piso – Edifício Bosque Oeste

lmarrone@info.unlp.edu.ar

Resumen: Los simuladores de red han mostrado ser una herramienta potente para múltiples fines. En particular para el estudio de redes, se ha desarrollado una herramienta de código abierto llamada Network Simulator (ns) que ha sido y sigue siendo de gran utilidad para el análisis de comportamiento de las redes. Si bien esta aplicación se ha utilizado en proyectos de investigación, también es una potente herramienta para la enseñanza en ámbitos universitarios y de posgrado. Con ella se puede determinar el comportamiento de una red, analizando parámetros típicos como tiempo total de transmisión, throughput promedio, gootput, ventana de congestión y muchos otros más. En el presente trabajo se expone el uso de la versión ns-2 como herramienta base para el desarrollo de pruebas de distintos tipos de topologías con variantes de protocolos TCP.

Palabras claves: SIMULACIÓN, TCP, PERFORMANCE, NS- 2.

Introducción

La simulación es una técnica utilizada en distintos ámbitos. En el estudio de las redes de computadoras es una práctica común el uso de esta potente técnica para comprender el comportamiento cuando se tienen distintas topologías, medios físicos de transmisión y protocolos. En particular el protocolo de comunicaciones TCP (Transmission Control Protocol) [1] ha sido ampliamente utilizado en las redes de datos desde su desarrollo a mediado de los años 70. Mediante modificaciones que tratan de adaptarlo a distintas condiciones de trabajo, ha acompañado a las innovaciones tecnológicas que se han desarrollado en el área de las telecomunicaciones y que utilizan tanto medios cableados como inalámbricos. Este protocolo se caracteriza por ser confiable, realizar un control de flujo y poseer un mecanismo de control de congestión de datos. También controla la secuencia de

entrega de los paquetes, mediante la verificación de recepción ordenada de dichos paquetes numerados en origen y verificados en el receptor. Así mismo ofrece un servicio orientado a conexión, que basa su entrega confiable en un procedimiento conocido como ARQ (AutomaticRepeatreQuest), en sus distintas variantes [2], que garantiza la integridad de los datos. Mediante el procedimiento ARQ y con la utilización de ACKs (acknowledgments) positivos, se logra que con menos de un ACK por paquete de datos se pueda confirmar la recepción de información de todo un conjunto de paquetes. Esta técnica se conoce como delayed-ACK [3] y permite lograr un importante aumento de eficiencia en el funcionamiento de la red.

A nivel de control de congestión, el protocolo TCP realiza una regulación del tráfico sobre el flujo de datos. Esta regulación se realiza verificando si existe una pérdida de segmentos o una recepción de ACKs duplicados. Si esto se verifica, el protocolo determina que existe una pérdida de paquetes y por consiguiente congestión en la red [4] [5]. Mediante el perfeccionamiento de este método, se ha desarrollado dos criterios para atender problemas de control de congestión. Uno de ellos se basa en un control reactivo del problema, suponiendo que existe congestión cuando ocurre una pérdida de segmentos. [6] [7]. Por otro lado, tenemos la otra variante que trata de realizar un control proactivo de la congestión, en donde lo que se trata es desarrollar una estrategia que permita evitar que el tráfico llegue a una situación de congestión [8] [9]. Para el estudio del comportamiento de estas estrategias, resulta evidente que poder contar con una herramienta que permita desarrollar pruebas en forma controlada, sin recurrir a costosos escenarios reales, brinda una comprensión acabada del comportamiento de un protocolo ante distintas situaciones, evitando un importante gasto de recursos.

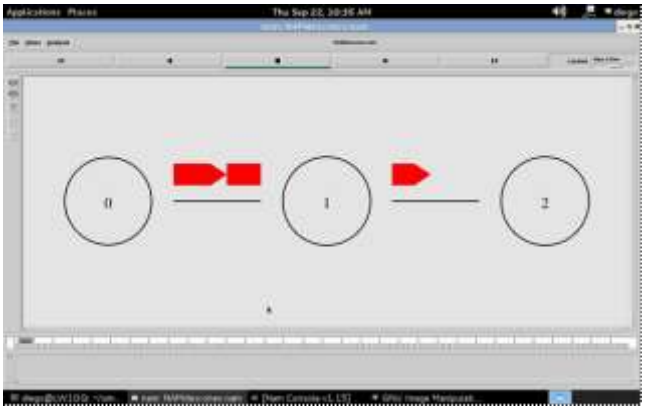
Esta herramienta fue desarrollada con este propósito y a modo de ejemplo presentamos la aplicación de la misma en un escenario en el que se evalúa el comportamiento de 3 implementaciones distintas de TCP, en un escenario de simulación de desconexiones de distintos tiempos, similar a lo que podría acontecer en una topología mixta con un enlace inalámbrico.

Cuando se diseñó e implemento el protocolo TCP los nodos que debía conectar eran fijos y no se contempló los inconvenientes típicos de una conexión móvil. Por lo tanto, las pérdidas de paquetes se debían casi exclusivamente a situaciones de congestión dado que las tasas de error de los paquetes en tránsito son muy bajas, características de los enlaces cableados. En una red inalámbrica las interferencias y demás características propias del medio producen pérdidas de paquetes en un grado mucho mayor que en el caso de redes cableadas. Frente a esa pérdida TCP reacciona interpretando la pérdida como una congestión en la red, cuando en realidad esta no ocurrió y dispara los algoritmos de control de congestión, reduciendo el flujo de paquetes inyectados. Las técnicas de control de congestión, se ha basado fundamentalmente en la detección de paquetes perdidos. Por ello, bajo estas condiciones, los protocolos reactivos entienden que hay congestión en la red y accionan sus algoritmos de control de congestión. Sin embargo existen situaciones en las cuales la pérdida de paquetes puede tener otro origen que no deberían disparar los mecanismos de control de congestión, lo que degrada fuertemente en rendimiento. En los enlaces inalámbricos las tasas de error son significativamente más altas y la presencia de desconexiones hace frecuentes las pérdidas de paquetes en tránsito. En base a esta respuesta es que nos interesa ver como desconexiones de distintos tiempo de duración afecta al flujo de datos, todo ello analizado con simulaciones realizadas con el ns-2 [10] [11]

Esquema de prueba

Para modelar y generar los datos del presente trabajo, se utilizó el ns-2 (Network Simulator 2), simulador de redes de eventos discretos en su versión ns-2.35 (released Nov. 4 2011). Si bien ya está disponible una nueva versión, el ns-3, optamos por ns-2 dado la gran cantidad de recursos de códigos (protocolos, aplicaciones) y la documentación disponible, en comparación con la versión ns-3.

Con este simulador se implementó la siguiente topología de 3 nodos:



Como vemos, se plantea este escenario simple para poder analizar el rendimiento del protocolo TCP frente a las desconexiones en los accesos con enlaces inalámbricos de un nodo móvil. Es por esta razón, que no se ha recurrido a las bibliotecas de ns- 2 destinadas a escenarios inalámbricos y se plantea un modelo con vínculos cableados donde se simulan los cortes. Como complemento y futuras aplicaciones de la herramienta consideraremos el papel del acceso inalámbrico en la performance de TCP.

La selección de este modelo, es una aproximación a un escenario wireless con un nodo fijo (nodo 0), una estación base (nodo 1) y un nodo móvil (nodo 2), con la simplificación práctica, que permite analizar el comportamiento de un TCP aislado frente a cortes o desconexiones de distinta duración.

Los nodos están vinculados por enlaces cableados. Ambos enlaces se configuraron como dúplex, con ancho de banda *1Mb/s* para cada uno de ellos, un retardo de propagación *10 ms.* y política de servicio de las colas *DropTail*.

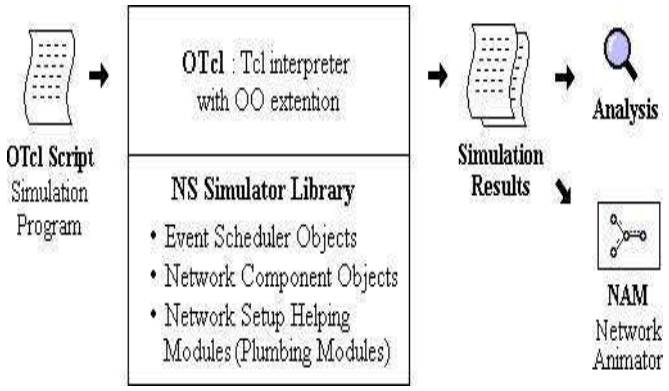
El nodo 0 se configuro como emisor y en él un agente TCP, por otro lado el nodo 2 se configuró como receptor. A este enlace se asoció un tráfico FTP (File TransferProtocol) como único tráfico.

Se realizaron 213 simulaciones independientes sobre las implementaciones de TCP. Para cada una de ellas, se corrió para los distintos tiempos de desconexión, con duraciones desde 0 hasta los 14 segundos, con pasos de 200 mseg. Los Agentes TCP que se utilizaron son: Linux, Reno y Vegas tal como están designados e implementados en esta versión de ns-2.

La duración en todas las simulaciones están condicionadas a la transmisión de 3.000 paquetes de TCP, de 1.000 bytes c/u. independientemente del tiempo de desconexión. Las desconexiones siempre comenzaron a los 13 segundos de iniciada la transferencia de datos y concluyó al terminarse de transmitir los 3.000 paquetes TCP asociado al tráfico de FTP.

Ns-2 está escrito en C++ y es de código abierto. Los scripts de interface de usuario utilizan el lenguaje OTcl que es una versión de Tcl orientada a objetos. Con este lenguaje se realiza la implementación del modelo, definiendo entre otros parámetros la topología, el tipo de protocolo TCP y las características del flujo de datos que se tiene.

Un esquema básico del simulador se observa a continuación:



Esquema básico de los componentes del ns2

Una vez diagramado el script que simula la red estudiada, se obtiene una representación ajustada a la red analizada. En esta simulación se puede agregar que los resultados sean volcados en un archivo de datos que luego es interpretado por el analizador NAm (Network Animator). NAm es una herramienta de animación de la red también basada en el lenguaje Tcl.

A continuación se presenta los componentes fundamentales del script utilizado para el presente trabajo.

```
#---------- DESCONEXIONES ----------------------------------------
set ns [new Simulator]                    CREA OBJETO SIMULADOR
#------- NAM trace file ---------------------------------------------------
set nf [open NAMdesconex.nam w]           APERTURA
$ns namtrace-all $nf                      ARCHIVOS
#------- NS trace file ------------------------------------ DE
set nt [open TRdesconex.tr w]             TRAZA
$ns trace-all $nt
```

```
#-----------------------------------------------------------------------------
#--- PROCEDIEMIENDO PARA FINALIZAR ------------------------------
procfinish {} {                          PROCEDIMIENTO
globalnsnf                               FINALIZAR
$ns flush-trace close $nf
exit 0
}
#--------- NODOS ------------------------------------------------------------
set n0 [$nsnode]                         CREACION DE
set n1 [$nsnode]                         LOS 3 NODOS
set n2 [$ns node]
#-------- ENLACES ----------------------------------------------------------
$ns duplex-link $n0 $n1 1Mb 10ms DropTail  CREACION DE
$ns duplex-link $n1 $n2 1Mb 10ms DropTail  LOS ENLACES Y CARATERISTICAS
#---------- TRAFICOS --------------------------------------------------------
##### TCP #####
set tcp0 [new Agent/TCP/Vegas]           CREA AGENTE TCP FUENTE
$nsattach-agent $n0 $tcp0
#-----------------------------------------------------------------------------
set sink1 [new Agent/TCPSink]            CREA AGENTE
$ns attach-agent $n2 $sink1              TCP DESTINO
#-----------------------------------------------------------------------------
set ftp0 [new Application/FTP]           CREA
$ftp0 attach-agent $tcp0                 APLICACION FTP
#-----------------------------------------------------------------------------
$ns connect $tcp0 $sink1                 CONEXION DE FLUJOS
#---- CORTE DEL LINK Y RECONEXION ------------------------------
$ns rtmodel-at 13.0 down $n1 $n2         CORTE ENLACE
#-----------------------------------------------------------------------------
$ns rtmodel-at 23.0 up $n1 $n2           CONEXION DEL ENLACE N1-N2
#------------ CWND del agente TCP ------------------------------------------
procplotWindow {tcpSourceoutfile} {
global ns                                ARCHIVO DE
set now [$ns now ]                       TRAZA DE
```

```
setcwnd [$tcpSource set cwnd_]                VENTANA DE
 # Agregartiempo                              CONGESTION
puts $outfile "$now $cwnd"
# [expr $now+0.01] 0.01 intervalo de muestra --
$ns at [expr $now+0.01] "plotWindow $tcpSource $outfile"
}
setoutfile [open "CWdesconex" w]
$ns at 0.0 "plotWindow $tcp0 $outfile"
#------------ TIMELINE ------------------------------------------------------------
$ns at 0.5 "$ftp0 produce 3000"               EJECUCION DE
#$ns at 51.5 "$ftp0 stop"                     ORDENES DEL SIMULADOR
#-------------------------------------------------------------------------------------
$ns at 52.0 "finish"                          LLANMDO AL PROCEDIMEINTO
#-------------------------------------------------------------------------------------
$nsrun                                        INICIO DELA SIMULACIÓN
```

Script OTcl – desconexiones.tcl

Para modelar y simular las desconexiones se utilizaron las siguientes instrucciones:

$ns rtmodel-at “tiempo 1” down $n1 $n2

$ns rtmodel-at “tiempo 2” up $n1 $n2

De esta manera se simula el corte del enlace que une el nodo 1 y el nodo 2 en el segundo 13 de iniciada la transferencia de información vía FTP y se restableció un tiempo después.

Después de ejecutada cada una de las simulaciones, se obtiene el archivo de traza, en donde están reflejados los eventos ocurridos. Este es un archivo de texto con el siguiente formato:

RESULTADOS OBTENIDOS

Tiempo total de transmisión

El tiempo total de transmisión es el tiempo medido desde que se comienza a trasmitir el primer paquete hasta que se recibe el ACK del último. Este valor se obtiene directamente del archivo de traza de cada una de las simulaciones.

Del archivo de traza se obtuvieron en forma directa, los valores de los tiempos requeridos, en cada simulación para completar la trasferencia de los 3.000 paquetes.

Las figuras 1, 2 y 3, se representan dicho tiempo para cada uno de los períodos de corte analizados, para los 3 agentes seleccionados (Linux, Vegas y Reno). Para el valor de coordenadas correspondiente a las abscisas, tenemos el tiempo de duración del corte y para ordenadas el tiempo total de transmisión. De esta manera, en 0 tenemos el ensayo para la simulación sin corte y en el otro extremo el valor del ensayo para el tiempo de desconexión máxima considerada, es decir 14 segundos.

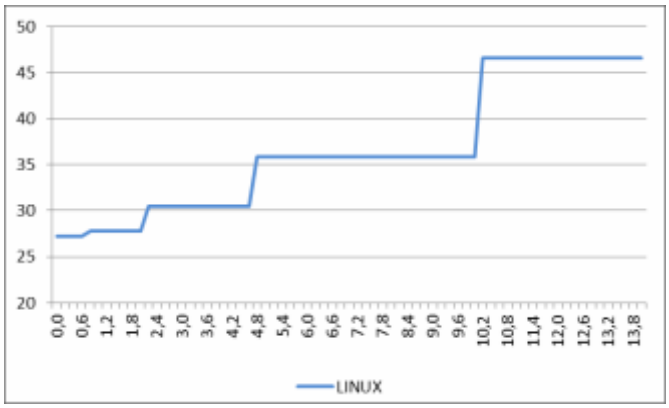


Fig. 1 – Tiempo total de transmisión Vs. Tiempo de desconexión (Linux)

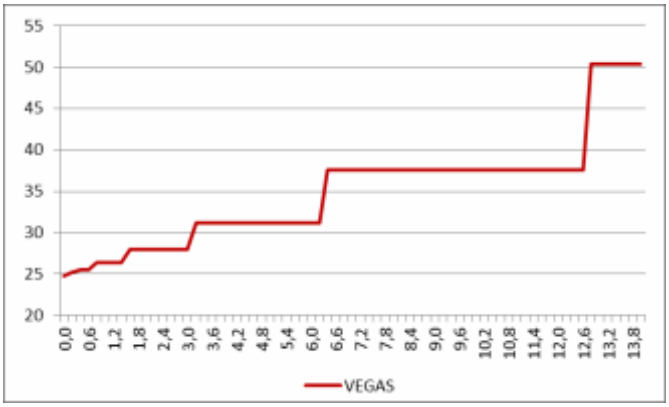


Fig. 2 – Tiempo total de transmisión Vs. Tiempo de desconexión (Vegas)

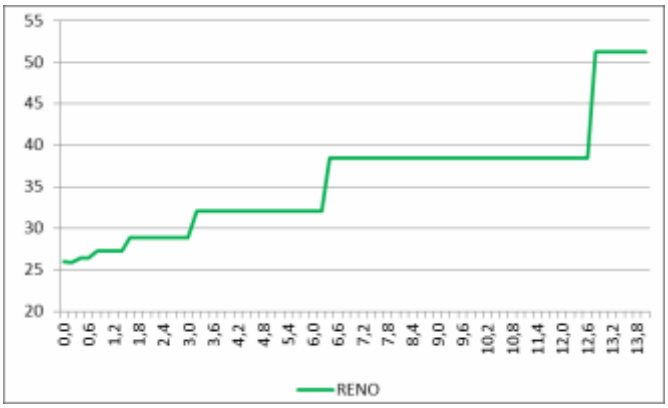


Fig. 3 - Tiempo total de transmisión Vs. Tiempo de desconexión (Reno)

Para tener una visión comparativa de la evolución de los valores se muestra en la figura 4, un gráfico con la confluencia de las 3 pruebas.

Como se observa el tiempo total de transmisión aumenta a medida que los valores de desconexión son mayores, lo que indica que los cortes degradan los tiempos de transmisión de datos.

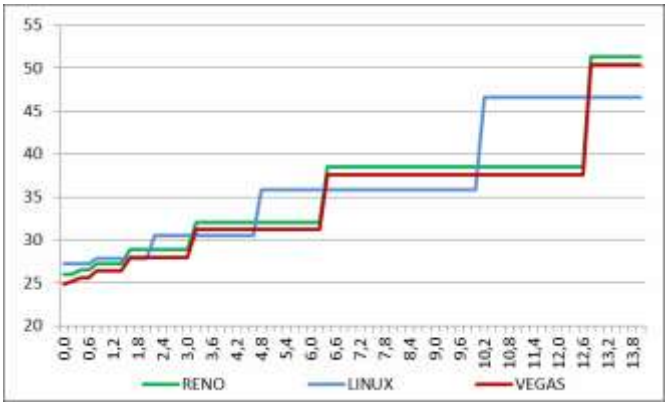


Fig. 4 - Tiempo total de transmisión Vs. Tiempo de desconexión

En esta representación vemos que los comportamientos de Vegas y Reno son muy similares, mientras que Linux, si bien tiene un comportamiento de incremento de tiempos de trasmisión ascendente no ajusta de la misma manera que los dos anteriores.

En la figura 4 se observa que, a medida que los tiempos de desconexiones se hacen mayores, el tiempos total de transmisión crece al mismo ritmo de crecimiento de los RTO (Retransmission timeout). Esto produce que la desconexión termine mientras el nodo emisor está aún con el timer activo. De esta forma, deberá esperar el tiempo

definido en el RTO, independientemente de la duración de la desconexión. Estos rangos de tiempos de espera se hacen mayores a medida que crecen los tiempos de desconexión, debido a que por cada timeout, el RTO se duplica. Esto genera los valores que describen un comportamiento amesetado (como se ve en la figura 4), pues durante ese intervalo no se produce un nuevo timeout. El tiempo total de espera, para la retransmisión, es idéntico para los distintos tiempos de desconexión comprendidos en ese rango. Esto produce dos fenómenos, por un lado que por más que el enlace vuelva a estar disponible, el emisor esperará un tiempo, que es igual al doble que el anterior, hasta que retransmite el segmento y por otra parte, más tiempos de desconexión estarán comprendidos dentro de este nuevo valor de RTO.

Throughput promedio

El throughput promedio se define como la totalidad de bytes enviados por el emisor dividido por el tiempo total que se necesita para que todos los datos sean enviados. Las unidades de medida están expresadas en Mbits por segundo (Mbits/s)

$$Throughput = \frac{Cantidad\ total\ de\ bytes\ enviados}{Tiempo\ total\ de\ transmision}$$

Para obtener los valores del throughput promedio para cada una de las simulaciones, se utilizó un script en Perl que los obtuvo a partir del archivo de traza. Se describe a continuación la forma de uso de dicho script:

Perl throughput<archivo traza><Nodo destino><Nodo.Puerto origen><Nodo.Puerto destino><granularidad>

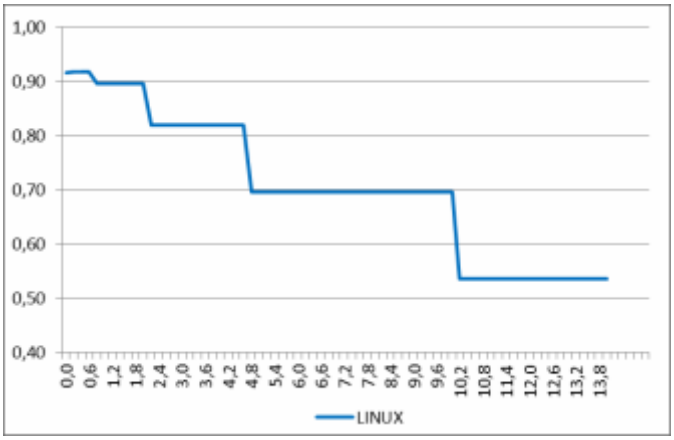


Fig. 6 – Throughput promedio para flujo TCP Vs. Tiempo de desconexión (Linux)

Fig. 7 – Throughput promedio para flujo TCP Vs. Tiempo de desconexión (Vegas)

Estos datos fueron procesados, volcados a una planilla de cálculo y se generaron los gráficos que aquí se presentan en la figura 6, 7 y 8. En ellas se observa que a medida que los tiempos de desconexión crecen, los valores de throughput promedio decrecen de manera sostenida. Al estar vinculado con la transferencia de datos, este parámetro se comporta como un negativo del tiempo total de transmisión.

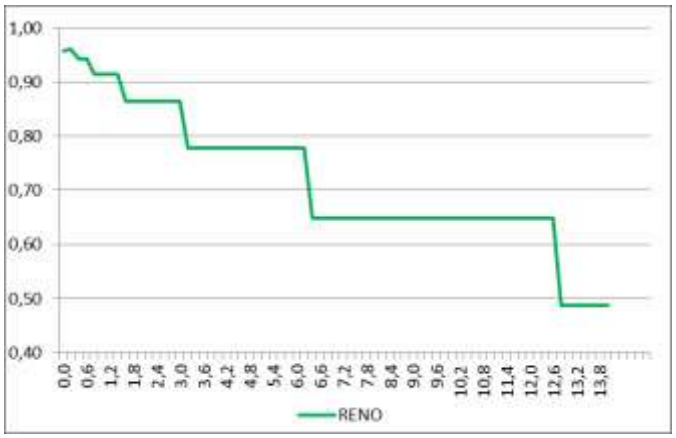


Fig. 8 – Throughput promedio para flujo TCP Vs. Tiempo de desconexión (Reno)

Al igual que en los casos anteriores se genera un gráfico en donde las 3 respuestas confluyen y muestran de manera simultánea los comportamientos de los tres agentes.

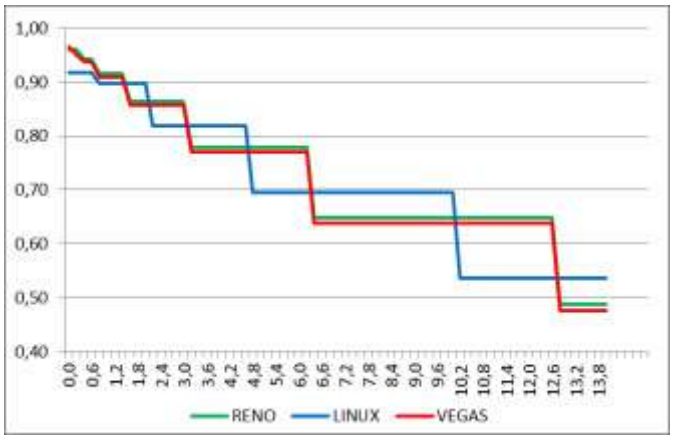


Fig. 9 – Throughput promedio para flujo TCP Vs. Tiempo de desconexión

En la figura 9 se observa en forma conjunta los tres comportamientos. Esto permite realizar una representación comparativa de cómo es la evolución del throughput promedio para los tres casos.

Como es lógico este valor registra un descenso paulatino de valores obtenidos, a medida que los valores de desconexión se incrementan, indiciado claramente una degradación de la calidad de la comunicación.

Goodput

El Goodput es el rendimiento a nivel de aplicación (es decir, el número de bits de información útiles suministrados por la red a un cierto destino por unidad de tiempo). La cantidad de datos considerados excluye los bits del protocolo así como los paquetes de datos retransmitidos. Esto se relaciona con la cantidad de tiempo desde el primer bit del primer paquete enviado (o entregado) hasta que se entrega el último bit del último paquete.

A partir de estos valores se define el Goodput Ratio y se calcula de acuerdo a la siguiente fórmula:

$$Goodput\ Ratio = \frac{bytes\ utilies\ para\ la\ aplicacion}{Total\ de\ bytes\ transmitidos\ por\ el\ emisor} \leq 1$$

Si expresamos este valor en función de los distintos valores de tiempo de desconexión para los tres protocolos analizados, tenemos las figuras que se observan a continuación.

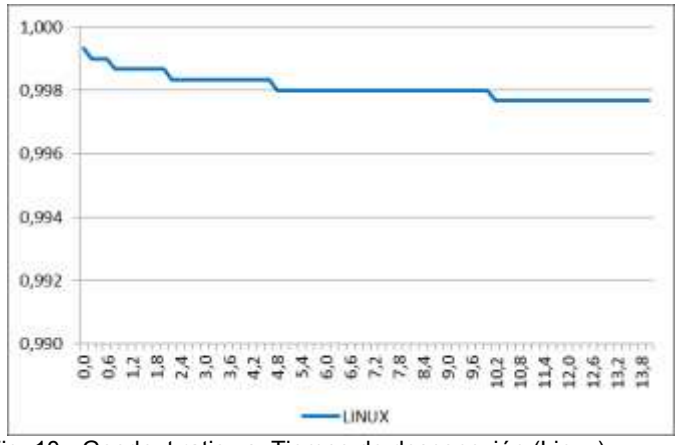


Fig. 10 - Goodput ratio vs. Tiempo de desconexión (Linux)

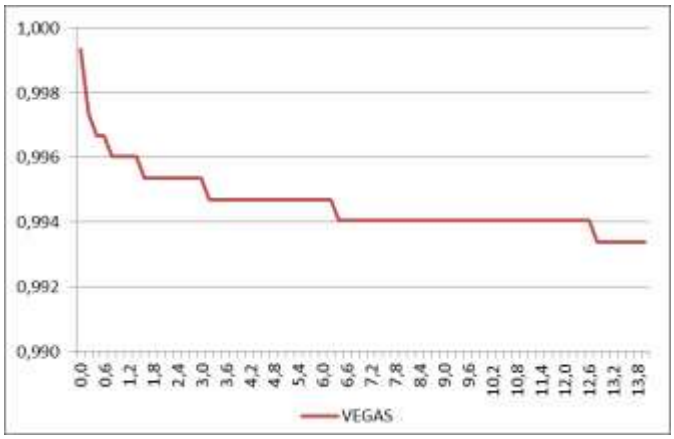


Fig. 11 - Goodput ratio vs. Tiempo de desconexión (Vegas)

En la figura 13 se integra el ensayo para les tres protocolos y permite una comparación relativa de los resultados

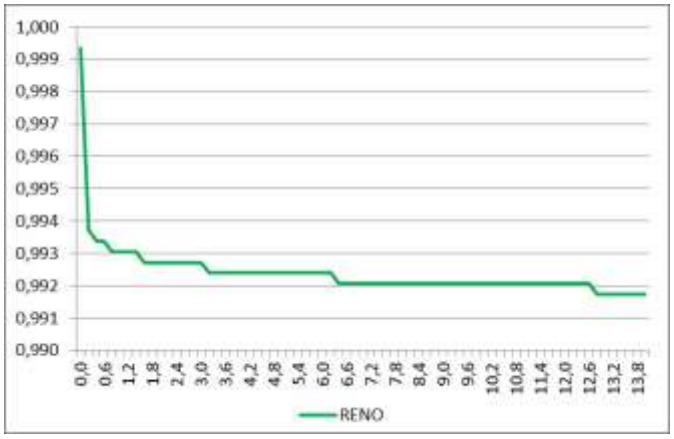


Fig. 12 - Goodput ratio vs. Tiempo de desconexión (Reno)

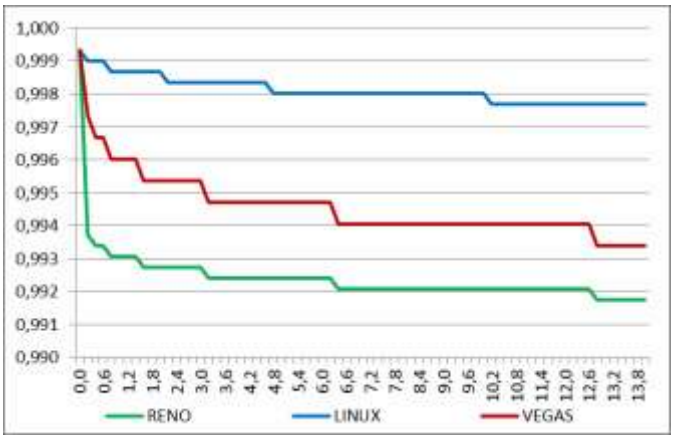


Fig. 13 - Goodput ratio vs. Tiempo de desconexión

Como se ve en la figura 13, la implementación con el agente Linux es la que se ve menos afectada ante los distintos tiempos de desconexión, manteniendo un comportamiento estable en su promedio de goodput ratio, que decae en el tiempo llegando a un valor mínimo de 0,9987, para un tiempo de desconexión de 14 segundos.

En el caso del agente Vegas, se observa que la caída del Goodput Ratio es más apreciable al comienzo y luego se mantiene relativamente estable, mientras que para el agente Reno se observa una fuerte caída relativa al principio y luego también estabiliza su comportamiento manteniendo una tasa de variación parecida al agente Linux para valores de tiempo de desconexión superiores 1,8 segundos.

Ventana de Congestion (CW)

En TCP, el tamaño de la ventana de congestión es uno de los factores que determina el número de bytes que pueden estar pendientes de transmisión en cualquier momento y en gran medida el rendimiento. Es el número de bytes que el emisor puede llegar a enviar sin esperar a recibir un ACK

La ventana de congestión es la forma en que el TCP emisor regula la inyección de paquetes en la red para evitar la congestión.

El objetivo de los mecanismos de control de congestión es lograr su valor óptimo, es decir, lo suficientemente grande de manera de lograr una buena velocidad de trasferencia pero no tan grande que pueda provocar congestión en la red degradando esa velocidad.

El tratamiento del ajuste de ajuste de la ventana de congestión, es una estrategia específica de cada protocolo. Los protocolos seleccionados paraesta prueba fueron determinados en buena parte por poseer estrategias notablemente distintas.

Para obtener el valor instantáneo del tamaño de la ventana, se incorporaron en el script OTcl del ns2 que define el modelo de red, líneas para que, en forma periódica (cada 0,01s), se almacene en un archivo (CWdesconex) el tiempo y el correspondiente tamaño de la ventana para cada una de las simulaciones.

Lo que se muestra en las figuras a continuación es un seriado de CW (ventana de congestión) Vs. Tiempo de duración de las desconexiones. El seriado corresponde para valores representativos que van de 0 a 10 segundos de duración. En la figura 14 tenemos una representación sin corte y lo podemos considerar un comportamiento base de referencia.

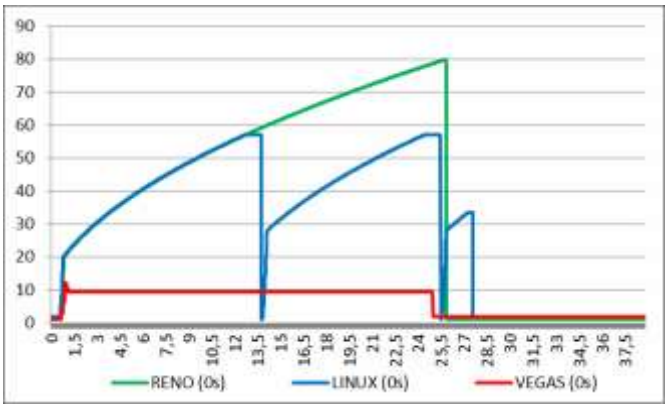


Fig. 14 - CW Vs. Tiempo (Sin desconexión)

En el otro extremo tenemos un valor de desconexión de 10 segundos, que representa a los valores de desconexiones de mayor longitud para este ensayo.

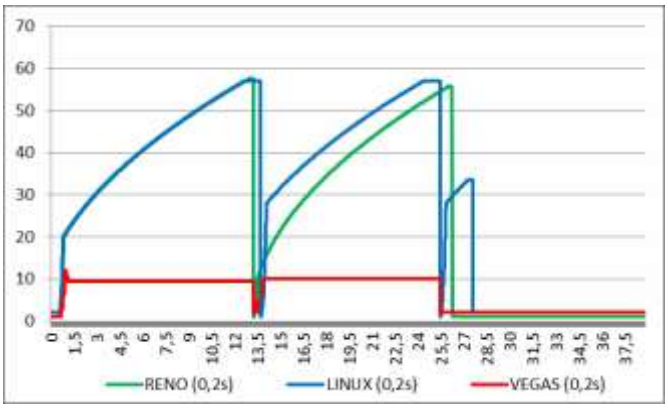


Fig. 15 - CW Vs. Tiempo (desconexión de 0,2 segundos)

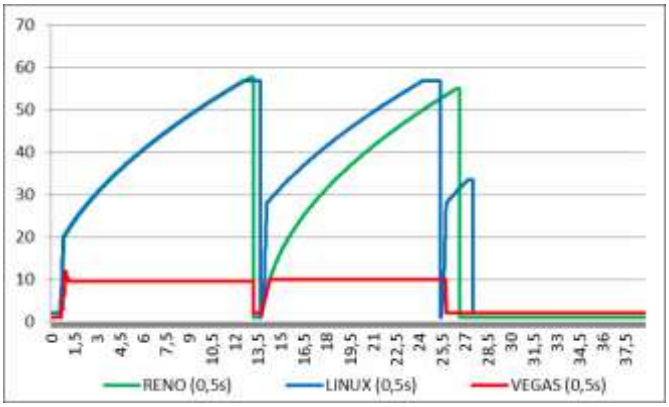


Fig. 16 - CW Vs. Tiempo (desconexión de 0,5 segundos)

Entre estos dos extremos, se muestran los valores más representativos que indican como los distintos algoritmos de las variantes del protocolo TCP ajustan su ventana de congestión. Es así que se representa la respuesta del modelo a un ensayo sin desconexión (0 segundos) y a 0,2, 0,5, 1, 5 y 10 segundos de tiempo de desconexión.

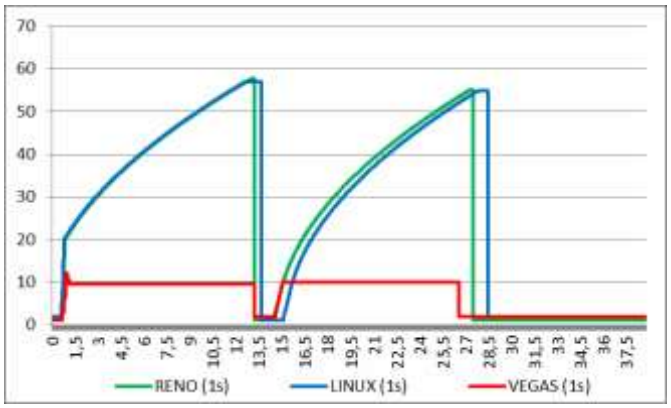


Fig. 17 - CW Vs. Tiempo (desconexión de 1 segundos)

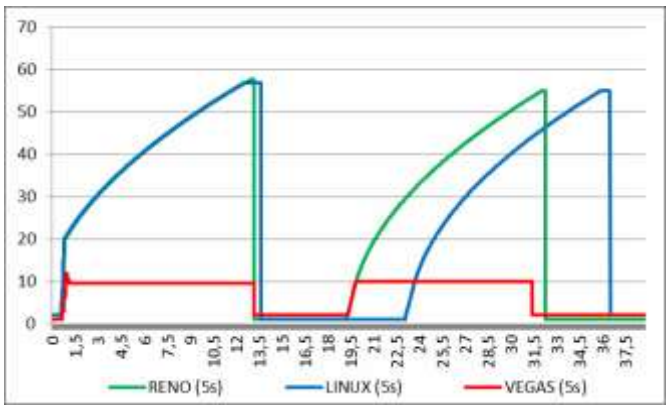


Fig. 18 - CW Vs. Tiempo (desconexión de 5 segundos)

En figuras 11, 12, 13, 14 y 15, vemos consecutivamente, como es el comportamiento cuando se introducen desconexiones de distintos tiempos de duración.

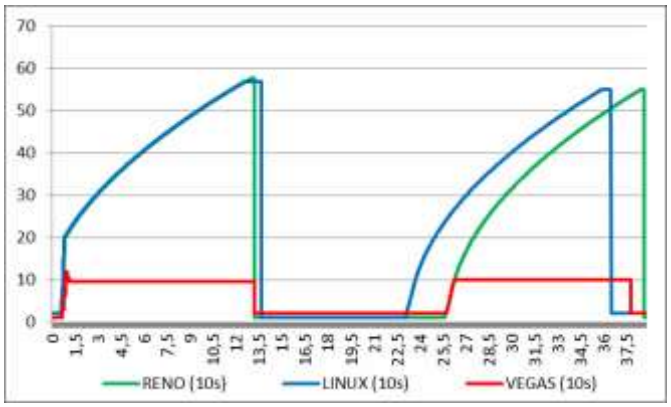


Fig. 19- CW Vs. Tiempo (desconexión de 10 segundos)

En este caso, a partir de los 13 segundos del ensayo, momento en el cual comienzan las desconexiones, se observa que los protocolos responden en forma similar, con una caída drástica de la ventana de congestión hasta llevar a su mínima expresión, manteniéndose constante en ese valor hasta que la desconexión finaliza. En ese momento reinician sus respectivas estrategias de crecimiento de la ventana de congestión, con el objeto de completar la transmisión del volumen restante de datos. Esto significa que se dispara el control de congestión de las tres implementaciones cuando se produce una desconexión a pesar de no existir congestión en la red, degradando el rendimiento del protocolo.

Reconocimientos

Este proyecto ha sido financiado con fondos para proyectos de investigación de la UNPA-UARG.

Conclusiones y líneas futuras de trabajo

Ns-2 muestra ser una robusta herramienta para analizar modelos de redes con una buena librería de variantes de protocolos de comunicación y mucho código abierto, desarrollado para obtener distintos tipos de resultados. Permite desarrollar distintas configuración de topologías y también se pueden desarrollar distintos escenarios donde recrear los posibles problemas que se dan en la realidad. El ejemplo desarrollado en este trabajo es solo una referencia de las posibilidades que la herramienta brinda. También existen librerías desarrolladas por usuarios con topologías básicas en donde se han recreado escenarios típicos.

Esta herramienta no solo se utiliza en ámbitos de investigación, también es utilizada en cátedras de grado y posgrado para ejemplificar la respuesta de una red.

En cuanto a las líneas futuras, se continuara el estudio con modelos wireless mediante la utilización de las librerías de NS-2, contemplando un modelo con todas las características inherentes a los enlaces inalámbricos y no solo el efecto de las desconexiones. De la misma manera, explorar otras variantes del protocolo, en distintos escenarios.

Referencias

[1] Postel J., "RFC 793: Transmission Control Protocol", September 1981.

[2] Stallings, William, "Comunicaciones y redes de computadoras". 6ta edición, Prentice Hall, 2000

[3] David Clark, RFC 813: Window and Acknowledgment Strategy in TCP, Julio 1982

[4] M. Allman, V. Paxson and W. Stevens, "RFC 2581: TCP Congestion Control", April 1999

[5] M. Handley, J. Padhye and S. Floyd, "TCP Congestion Window Validation", RFC 2861, June 2000.

[6] Allman M., Paxson V. and Blanton E., "TCP Congestion Control", IETF, Standars Track RFC 5681, Sept. 2009.

[7] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, "Host-to-Host Congestion Control for TCP," IEEE Communications Surveys Tutorials, vol. 12, no. 3, 3rd quarter 2010, pp. 304- 340.

[8] Stevens, W., "TCP slow start, congestion avoidance, fast retransmit, and fast re-coveryalgorithms".RFC 2001, 1997.

[9] V. Jacobson, "Congestion Avoidance and Control," ACM SIGCOMM Computer Communication Review, Vol. 25, No. 1, pp. 157-187,1995.

[10] Teerawat, Issariyakul&EkramHossain, "Introduccion to Network Simulartor NS2", Springer, 2009

[11] The ns Manual, (formerly ns Notes and Documentation). A Collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC. Kevin Fall &KannanVaradhan, Editores, Nov 5, 2011 (ns-2.35)